Interaction Techniques for Large Directed Graphs

Rudy Darken
Manuel Perez

The George Washington University
Department of Electrical Engineering and Computer Science
Washington, D.C.
darken | manuel @seas.gwu.edu

January 16, 1992

Abstract

This paper describes interaction techniques for large directed graphs which enable fast and efficient browsing and editing. A paradigm of views into virtual graph space is employed. The basic concept is to allow users to view their positions from global and local perspectives simultaneously. With effective linking between global and local views of the graph, navigation problems can be minimized. In addition to a description of a solution incorporating this methodology, examples will be given of past solutions to the graph navigation problem.

Introduction

Graph navigation is a recurrent problem faced by any system which may have to manage graphs having hundreds or thousands of nodes. When a graph becomes very large, it can either be displayed in its entirety making details too small to be usable, or it can be displayed in pieces giving the user no feedback on global position or overall graph topology. Historically, the solution has been to merge these two methods in such a way that a section of the graph is displayed in a workable scale and a global map is used to show the graph as a whole and the user's position within it [DONE78].

The Graph Browsing/Display (GBD) system will be described which provides users with the ability to browse, edit, and navigate within very large graphs in an efficient manner. The method employs a paradigm of views into virtual graph space. The basic concept is to allow users to view their positions from global and local perspectives simultaneously. With effective linking between global and local views of the graph, navigation problems can be minimized. In addition to a description of the GBD system which incorporates this methodology, examples will be given of past solutions to the graph navigation problem.

Key Issues & Requirements

Inherent to any solution to the graph navigation problem is a method of movement which is intuitive yet powerful. Since the graph is very large, it is likely that without guidance, a user may become disoriented and be unable to function properly. For this reason, a solution must always show a user his *absolute* location in the graph. This view is unchanging and will anchor a user's sense of location to a fixed point. Also, there must be a way to select a part of the graph of interest and be able to work on a view of workable scale.

The GBD system was designed for use with a specification language for real-time systems which displays its data as a large, nested, directed graph in which children of a given node are represented as nested boxes or subgraphs within the parent [JAHA88]. Clearly, this data is structured and the interface should take advantage of that structure. Figure 1 shows an example node in such a graph. There is no specified limit to the depth of this hierarchy. This property gives these graphs a three dimensional quality. Associated with these types of graphs is a new requirement of zooming into the graph to enlarge lower level information to a workable size. Also, users may be browsing these graphs with the intent of making changes to them. A WYSIWYG direct manipulation interface employing methods familiar to users is best suited for this type of environment. Editing functionality must include adding, deleting, and moving nodes and edges. Along with editing, the user must be able to layout a graph using a prescribed layout program.

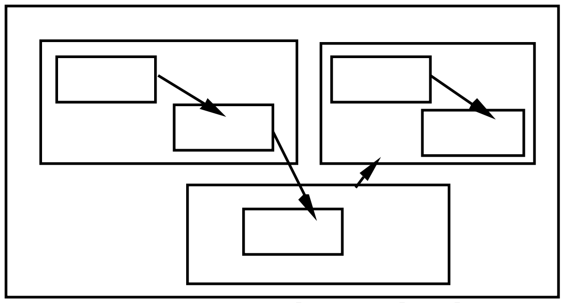
Figure 1: Example nested node.

It is assumed that graphs may be as large as 10^4 nodes. With multiple users working on the same data, how does the system restrict access to information so as not to allow overwriting or deleting another user's work? How can a user be informed of another user's position in the graph?

When a user is working with a very large graph, there is always the threat of overflow of information. The user is presented with much more information than can possibly be comprehended. The interface must allow the user to hide distracting information when requested and have knowledge as to what information to show and at what level of detail to show it.

A Survey of Previous Solutions

It has been evident from the beginning that the only way to effectively interact with a large graph is to view an enlarged part of the graph and the whole graph at the same time. But what is the best way to do this? The following are descriptions and evaluations of relevant work previously done in this area which try to answer this question.

Generalized Fisheye Views --- 1986

A fisheye view uses a "degree of interest" function to determine what should be displayed in a complex graph or space. The function states that the degree of interest the user has in viewing a certain object increases with object importance and decreases with distance from the user. The importance and distance factors might be described as size, level in the hierarchy, or some other characteristic of objects in the space or graph. The end effect is that detail is only present in the position where the user presently is browsing. Only the most important objects are shown at a distance [FURN86].

This method assumes that nodes closely related to each other will be locally spaced. This is not necessarily the case. The relation between any two nodes is at the discretion of the user. There may be many ways to rate importance among nodes based on what the user might be interested in. In essence, the degree of interest function is likely to change for each user. This is unacceptable for nested graphs. Multiple users must see the same graph from their own perspective. This would change the graph entirely. Also, based on what the degree of interest function was chosen to be, the view might still be far too cluttered with information.

A Browser for Directed Graphs --- 1987

This method concentrates on the graph layout issue. It was determined that graph layout should not be called automatically but only when one or more of the following conditions are met [ROWE87]:

- The user explicitly requests a new layout.
- A particular subgraph becomes too complicated.
- The subgraph partition must be changed. Either a set of nodes are to be moved or a subgraph is to be partitioned.

If the criteria for deciding when a graph has become too complicated varies for different users, then this method becomes ambiguous. Also, if a graph is partitioned or a set of nodes are moved, it might have been because this was more readable to the user. It would be unreasonable to have the system automatically layout the graph under any circumstances unless it was found to be valid and beneficial in all cases.

The navigational technique employed was not designed explicitly for very large graphs and thus is primitive in that it only allows horizontal and vertical scrolling and zooming but has no global view of the graph space. Due to its lack of a global map, users will be prone to becoming disoriented in the graph space. This technique becomes less efficient as the graph grows. Editing functionality includes inserting, deleting, moving or changing nodes or edges.

The Tourist Artificial Reality --- 1989

A technique is introduced which simulates a physical space when dealing with a large virtual workspace such as graphs, trees, or hypertext. Consistency is enforced on the space. Changes made by one user must be evident to another user viewing the same space. It is not required that all users get the same view of the same data, however. Users are given a view specialized to their needs. For example, programmers, managers, and end users see a different form of the data. A constraint is placed on the amount of information which is allowed to be displayed to any user. This is achieved using filters which block irrelevant information from being displayed to a user who is not interested in that information [FAIR89].

This paper offers several methods of solving problems related to multiple users. A method is described in which users are represented in the view along with other users sharing the same data space. The problem of user conflict within a data space was neglected. Can a user manipulate data which another user is presently sharing? The notion of altering the interface for different users can be applied to some applications but not necessarily to all. Much of the information filtering that this system does is automatic. For example, if the system knows that I am a manager, then I can only view the manager's interface. I do not have access to the coder's interface or the data associated with it. For many applications, this is unacceptable. It may be advantageous to hide information at a given time from a given user but he should never be denied access to it if it is requested.

A Browser for Large Directed Graphs --- 1990

This is the only solution to the graph navigation problem to be discussed which is specific to large graphs. It was applied to graphically representing a Unix file hierarchy in graph form. Windows have variable magnification. The view can be panned or zoomed. Global window location is shown in all windows it appears in. Figure 2 shows an example in which a window is represented within another window. A window may be moved quickly by grabbing its rectangle from within another window and moving it to the desired location [BOVE90].

Another difficult problem is that of node and link visibility. When does a node or link appear and when is it hidden? A function was used based on the size of the node and the window magnification which would determine node visibility. Visibility of links can be selected based on given criteria. If this criteria is dependent only on the readability of the graph, then it will be independent of the application and will always provide the interface with a cleaner and less cluttered view. If this is not the case, then the user may find information being hidden from view without knowledge of its existence.

Although this solution handles more of the problems to be encountered with large graphs, the method of displaying the global view might allow users to become disoriented easily since it does not anchor any view to a fixed zoom factor.

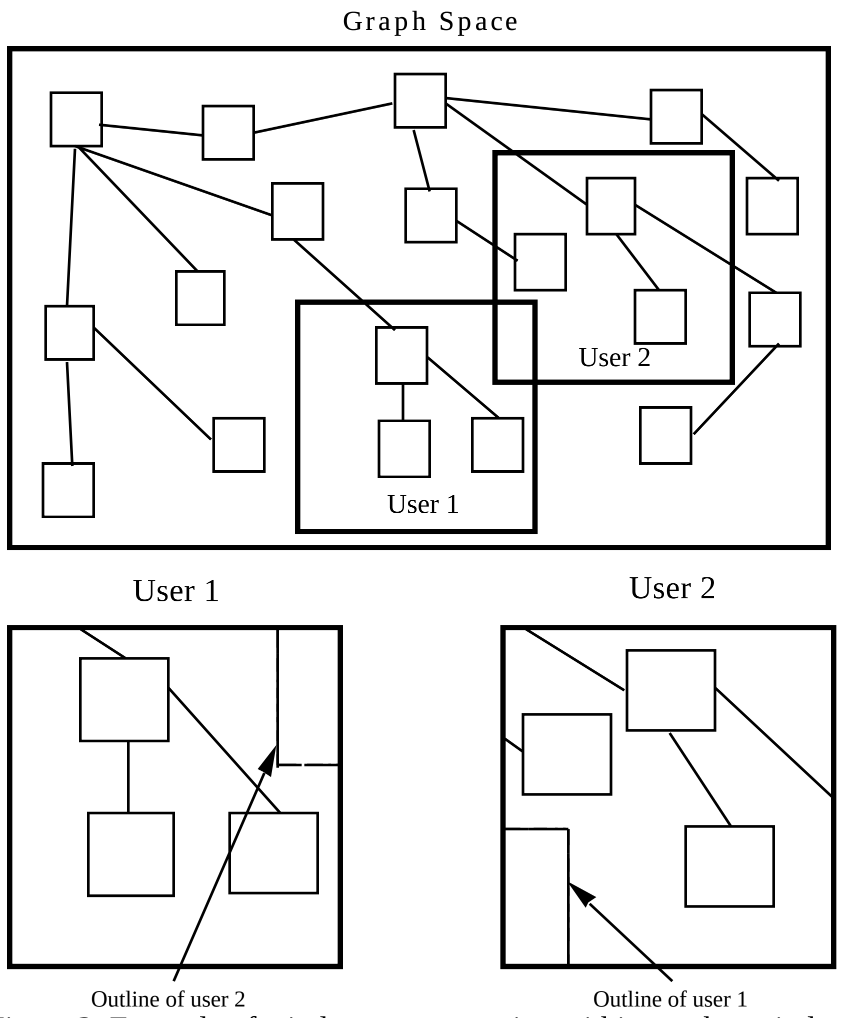

Figure 2: Example of window representation within another window.

Navigation Solution

The design for the GBD user interface is a hybrid of many of the systems previously described. Some of the interaction techniques are designed explicitly for use on nested, directed graphs but as a whole, these methods can be applied to any large graph.

Windows

There are two types of windows associated with the GBD user interface:

•*work* windows -- An enlarged portion of the graph.

•*locator* window -- A global map of the graph.

The Work Window

The GBD user interface employs a paradigm of views into virtual graph space. Work windows provide the views and the ability to navigate through graph space. In addition, they also have links to the locator window's global map in order to help the user to be aware of his present position in space. A work window is intended to be a workspace which provides maximal flexibility to the user in allowing editing and simple navigation. There is no limit to the number of work windows the user may have in any graph space.

Interaction with a Work Window

Upon creation of a work window, it is given a name, a default size and placed in a default position in the graph. Resizing does not scale the contents but rather enlarges the view into the graph space. Actions involving the size, location or zoom factor of a work window will effect the window's corresponding *indicator* in the locator window. Indicators will be described in detail later. Scrollbars are present to translate the work window over the graph space. This is fine movement as opposed to coarse movement which can be done from the locator window. The user may zoom in or out causing the contents of that window to enlarge or reduce. Figures 3 and 4 show an example of a graph space and the state of the work window and its indicator before and after a zoom. Since the view is of a dynamic size, the graphs may experience growth limited only by the hardware restrictions of the system.

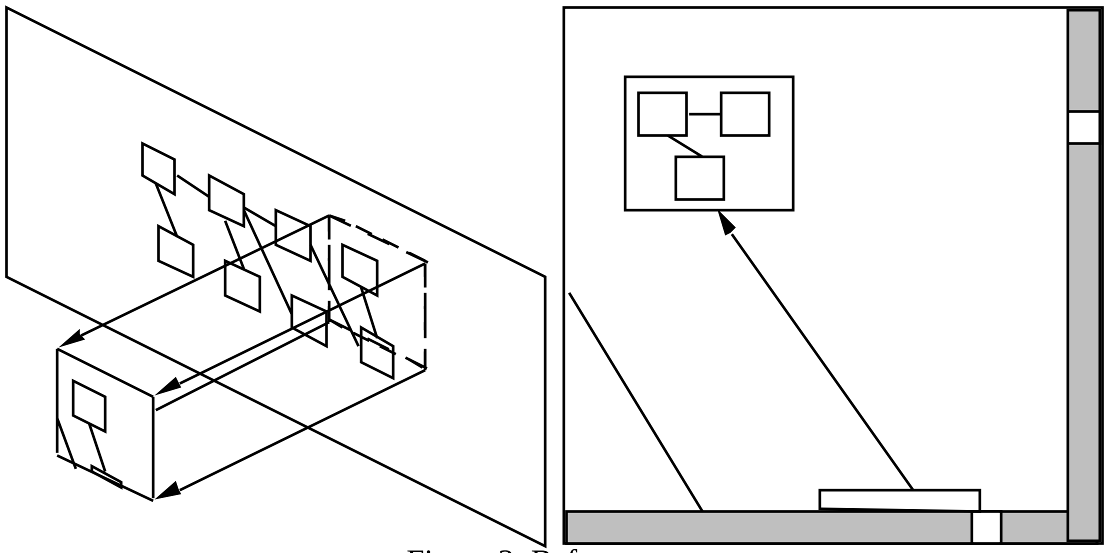
Figure 3: Before zoom.

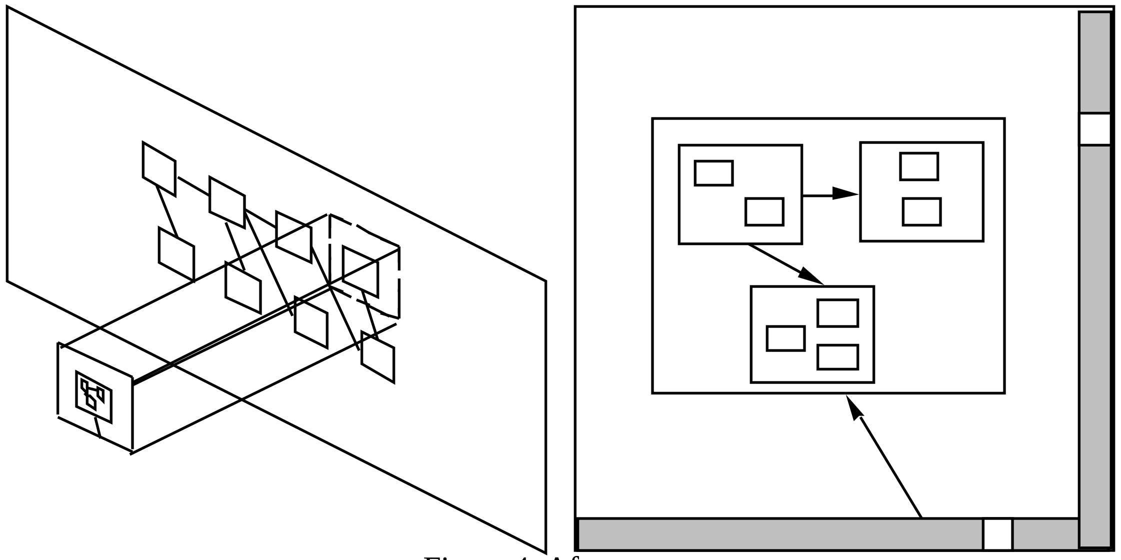
Figure 4: After zoom.

In addition to movement and navigation, from each work window, the user can modify the data presently in the window. All editing functions are performed in a direct manipulation fashion similar to typical drawing programs. New objects can be created by selecting an object type (node or edge) and drawing it directly in the workspace. Text such as node labels or tags can be added or changed. Existing objects are moved by selecting them and dragging them to a desired location. The typical edit functions of cut, copy, paste, delete, and duplicate are also supported as well as a clipboard and operate as expected. All editing is done within a work window but items may be moved from window to window using the clipboard. The interface also allows the user to have multiple graphs open which enables items to be moved or copied from one graph to another.

If a user wishes to conduct a search for a specific keyword or label given to a node or edge, a find operation is provided which, if successful, moves the window to be centered over that object. Partial searches are acceptable.

There exists a threshold zoom factor after which the system will iconify text associated with objects in the graph. This is done to unclutter the workspace. The icon is given to cue the user that text does exist for the object but is hidden from view. There are possible future considerations involving hiding and showing data in the graph which are discussed later.

Whenever necessary, the user can automatically layout the graph. Graph layout is never done without being explicitly requested by the user. The system makes no assumptions as to why nodes and edges are presently placed. The user is able to ensure that data is local to related data. The term "related" is defined by the user. To achieve this end, the user may request that only a certain level of the graph hierarchy be laid out or only a small portion of the graph. This flexibility allows the user to maintain the topology of the graph in an efficient way.

The Locator Window

There is only one locator window associated with the interface. This window is the global map of the graph. It will always show the whole graph, therefore there is no scrolling or zooming on this window. Its purpose is to display the overall structure and topology of the graph to the user.

Interaction with the Locator Window

The locator window can be resized which will scale its contents to fit in the window. When a graph becomes very large and its topology is unclear from the locator view, it can be scaled to increase resolution of the graph.

Linking the Work Windows to the Locator Window

Rectangles called *indicators* are drawn in the locator window to show the relative position and size of work windows in the graph. When a work window is moved its corresponding indicator moves and when a work window is resized or zoomed, its corresponding indicator is resized. See Figures 3 and 4.

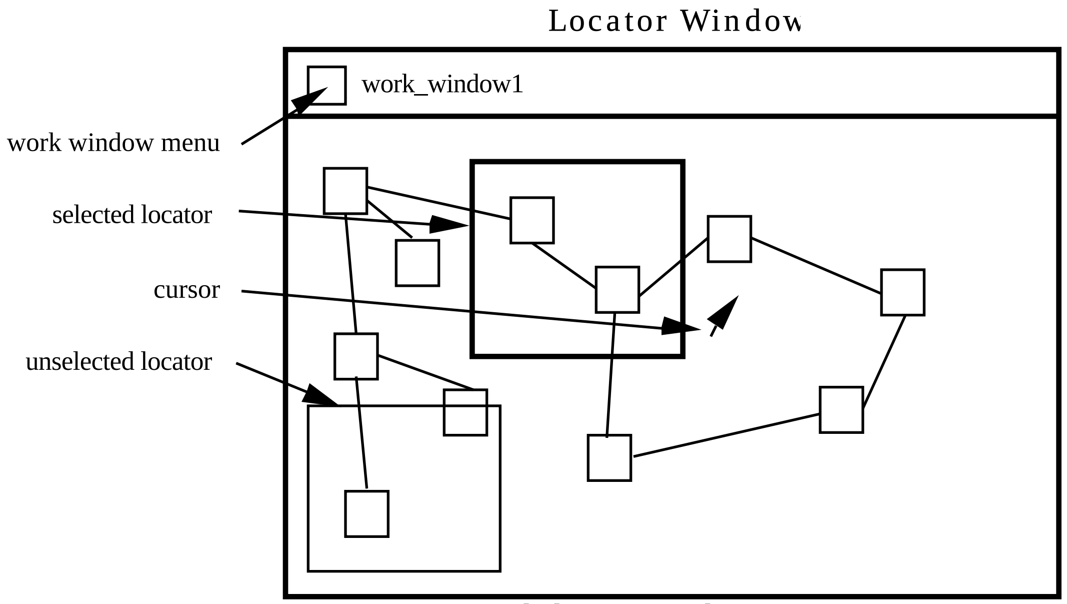

Figure 5: Example locator window.

Along with the graph map in the locator window, there is also a work window menu which names the work window which is presently active or selected. A hooking algorithm is used [ ] so that when the mouse cursor enters the locator window, an indicator is always selected and its name shown in the work window menu. Figure 5 shows an example locator window. Note that "work_window1" is presently selected and is represented as a bold rectangle. It has been selected by the hooking algorithm since the cursor is closer to it than the other indicator. When a window is selected, its indicator is highlighted and its name is in the work window menu. A user can use this feature to quickly select a number of work windows to determine which is of a particular name. If the user wishes to make this window active, he can either click on its indicator, or select its name from the work window menu. When a work window is active, it is brought to the top of the desktop, its indicator is highlighted in the locator window, and its name is placed in the work window menu. This is a method of linking the locator window to working windows. Coarse movements can be made by clicking on an indicator and dragging it. On release, that work window will be updated to reflect its new location. Finer movements can be made using the scrollbars on the work window itself.

Future Research

It is uncertain what would be the best method to hide data from the user to unclutter the workspace. Clearly, it is advantageous to be able to remove information which a user is not interested in and which might become a nuisance. As with graph layout, since the interface is generic and makes no assumptions about the type of graph being viewed, information should never be automatically removed from view. An exception is made when the system iconifies text when it becomes too small to be readable. What would be useful to be able to remove? And if it were removed, should it be completely removed or iconified? If it is removed completely, a user may become unaware of its existence. If it is iconified, a view may still have a cluttering problem. This will be implemented and tested in the future.

The other area of concern is in handling the multiple user problem. Any project involving large graphs must have many potential users. They should be able to work on the graph simultaneously. Work windows of concurrent users should be represented in some way. It has not been determined at this time whether it is more effective to show this in the work windows or in the locator window or both. We want to avoid cluttered displays but if the multiple user problem becomes an important issue, it may be valid to show users in work windows. Furthermore, if this information is to be displayed, it must be updated whenever a user moves, resizes, or zooms a work window. Also, the graph itself must be updated to reflect recent edits. This implies that all users are networked together and this information is broadcast to all other users when necessary.

If multiple users are to be allowed concurrent access to the graph, there must be a record locking scheme to avoid problems with overwriting data. A possible method might be to allow only one user read/write access to any node at one time. That is, the first user to view a selected portion of the graph will have read/write access while all subsequent users will have only read access. This is a simple solution which needs further refinement.

Conclusion

Although this design was made specifically for nested, directed graphs, it addresses the problems specific to very large directed graphs in general. The linking between the global view and the working view is clean and consistent. The metaphor of views into virtual space aids the user in visualizing the graph and how to interact with it.

In the future, these ideas will be further refined through prototype testing and iterative alteration to the design. It is apparent that there are parts of the design which may prove unacceptable. But the foundation of the design is built upon successful past work in the area of graph navigation and we expect that the GBD user interface design will make further improvements on this past work.

References

[BOVE90] Bovey, J.D. "A Browser for Large Directed Graphs," *Computer Graphics Forum*, September 1990, 195--204.

[CHEN91] Chen, D. "An Automatic Layout Module for Modechart," *University of Texas Technical Report*, April 1991.

[DONE78] Donelson, W. "Spatial Management of Information," *Computer Graphics (Proc. ACM SIGGRAPH 1978)*, 12:3, August 1978, 203--209.

[FAIR89] Fairchild, K., Meredith, G., and Wexelblat, A. "The Tourist Artificial Reality," *Proc. ACM SIGCHI 1989*, May 1989, 299--304.

[FEIN88] Feiner, Steven. "Seeing the Forest for the Trees: Hierarchial Display of Hypertext Structure," *Proc. COIS 1988 (Conference on Office Information Systems)*, 1988, 205--212.

[FURN86] Furnas, George W. "Generalized Fisheye Views," *Proc. ACM SIGCHI 1986*, April 1986, 16--23.

[HAMM88] Hammond, N., and Allinson, L. "Travels Around a Learning Support Environment: Rambling, Orienteering or Touring?" *Proc. ACM SIGCHI 1988*, May 1988, 269--273.

[JAHA88] Jahanian, F., and Mok, A. "Modechart: a Specification Language for Real-Time Systems," *21st Hawaiian Conference on System Sciences*, January 1988.

[KAMA89] Kamada, T., and Kawai, S. "An Algorithm for Drawing General Undirected Graphs," *Information Processing Letters, 1989*, vol. 31, 7--15.

[ROWE87] Rowe, L., Davis, M., Messinger, E., Meyer, C., Spirakis, C., and Tuan, A. "A Browser for Directed Graphs," *Software--Practice and Experience*, 17:1, January 1987, 61--76.

[SMIT87] Smith, R.B. "Experiences With the Alternate Reality Kit--An Example of the Tension Between Literalism and Magic," *Proc. ACM SIGCHI 1987*, April 1987, 61--67.